1. What is a buffer?

A **buffer** is a contiguous block of memory allocated to temporarily store data during program execution. Buffers are commonly used to hold data like text, numbers, or even raw binary data while the program processes it. They act as a *staging area* for inputs and outputs.

Key Characteristics of a Buffer:

1. **Fixed Size**: Buffers are allocated a specific size in memory (e.g., 10 bytes or 256 bytes).
2. **Data Storage**: Buffers temporarily hold data during operations like reading user input, file I/O, or networking.
3. **Access Mechanism**: Data in a buffer can be accessed using memory addresses or array indices.

Examples of Buffers:

1. **Input Buffer**:
   When a program reads user input, it uses a buffer to store the input temporarily before processing it.
   Example: A buffer to hold text entered by a user in a form.
2. **Output Buffer**:
   When a program writes data to a file or sends it over a network, the data may first be placed in an output buffer.
   Example: Buffering data before sending it as a network packet.

Analogy:

Think of a buffer like a bucket:

- It has a specific capacity (size).
- You can fill it with water (data).
- If you pour too much water (data), it overflows, spilling into surrounding areas (adjacent memory). This is where buffer overflows occur.

Buffer in Code:

Here's an example of a buffer in C:

```c
char buffer[10];  // Allocates a buffer of 10 bytes to store text
```

You can fill this buffer with up to 9 characters (leaving space for the null terminator `\0`):

```c
strcpy(buffer, "Hello");  // Safe: fits within the buffer
strcpy(buffer, "Hello, World!");  // Unsafe: overflows the buffer
```